The CPU Performance Equation

Prof. Naga Kandasamy
ECE Department
Drexel University

March 29, 2016

Computers are synchronous machines in that their operations are controlled by a clock running at a constant rate. So the CPU time consumed by a program can be obtained as:

$$\text{CPU time} = \text{CPU clock cycles for the program} \times \text{clock cycle time}$$

In addition to the number of clock cycles needed, we can also count the number of instructions executed by the program—the instruction count (IC). If we know IC and the number of clock cycles, we can then calculate the number of cycles per instruction (CPI):

$$\text{CPI} = \frac{\text{CPU clock cycles for the program}}{\text{IC}}$$

CPI is a key figure of merit that is used to characterize different styles of instruction sets and implementations.[1] So the CPU time can be written in terms of the CPI and IC as

$$\text{CPU time} = \text{CPI} \times \text{IC} \times \text{clock cycle time}$$

Sometimes it is useful to calculate the number of processor clock cycles as

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{CPI}_i \times \text{IC}_i,$$

where IC_i is the number of times instruction i is executed in the program and CPI_i represents the average number of clock cycles needed to execute instruction i. The CPU time can then be calculated as

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{CPI}_i \times \text{IC}_i \right) \times \text{clock cycle time}$$

The average CPI can be calculated as

$$\text{CPI} = \frac{\sum_{i=1}^{n} \text{CPI}_i \times \text{IC}_i}{\text{Instruction count}} = \sum_{i=1}^{n} \frac{\text{IC}_i}{\text{Instruction count}} \times \text{CPI}_i$$

[1] Designers also use the inverse of CPI which is instructions per cycle (IPC).

Let us look at a few examples detailing the use of the CPU performance equation.

Example 1. Suppose we are considering two alternatives for a conditional branch instruction, as follows:

- CPU A: A condition code is set by a compare instruction and followed by a branch that tests the condition code.
- CPU B: A compare is included in the branch statement itself.

On both CPUs, the conditional branch instruction takes 2 cycles, and all other instructions take 1 clock cycle. On CPU A, 20% of all instructions executed are conditional branches; since every branch needs a compare, it follows that another 20% of the instructions are compares. Because CPU A does not have the compare included in the branch, its clock cycle time is 25% faster than CPU B's. Which CPU is faster?

Let us first determine the CPI of CPU A as

$$\text{CPI}_\text{A} = .20 \times 2 + .80 \times 1 = 1.2,$$

since 20% are branches taking 2 clock cycles and the rest of the instructions take 1 clock cycle. The performance of CPU A is then given by

$$\text{CPU time}_\text{A} = \text{IC}_\text{A} \times 1.2 \times \text{Clock cycle time}_\text{A} \tag{1}$$

Let us now calculate the CPU time for CPU B. Note that compares are not executed on this CPU, and so the number of instructions to be executed by CPU B, relative to CPU A, is

$$\text{IC}_\text{B} = 0.8 \times \text{IC}_\text{A},$$

since compares were 20% of the original code. So, the frequency of branch instructions in the new code is $\frac{20\%}{80\%}$, or 25% of the instructions in the new code are branch instructions, taking 2 clock cycles, while the remaining 75% of the instructions take 1 clock cycle. So

$$\text{CPI}_\text{B} = .25 \times 2 + .75 \times 1 = 1.25.$$

Also,

$$\text{Clock cycle time}_\text{B} = 1.25 \times \text{Clock cycle time}_\text{A},$$

since the clock on CPU A is 25% faster than the one on CPU B. The performance equation for CPU B is given by

$$\begin{aligned} \text{CPU time}_\text{B} &= (.80 \times \text{IC}_\text{A}) \times 1.25 \times (1.25 \times \text{Clock cycle time}_\text{A}) \\ &= 1.25 \times \text{IC}_\text{A} \times \text{Clock cycle time}_\text{A} \end{aligned} \tag{2}$$

Comparing (1) and (2), CPU A with the shorter clock cycle time is faster than CPU B which executes fewer instructions.

Example 2. Suppose we are considering a change to the instruction set. Assume that the processor has only loads and stores to memory, and then all operations work on the registers—much like

the MIPS processor (from ECEC 355). Now, assume that you have profiled the code and the instruction mix is detailed in Table 1. Let us now assume that 25% of the ALU operations directly use a loaded operand from memory that is not used again. As a designer, you propose adding ALU instructions that have one source operand in memory. These new "register-memory instructions" have a CPI of 2. The extended instruction set increases the CPI of branches by 1, but does not affect the clock cycle time. Would this change improve CPU performance?

Table 1: The instruction mix and CPIs of individual instructions

Operation	Frequency	CPI
ALU Operations	43%	1
Loads	21%	2
Stores	12%	2
branches	24%	2

We use the CPU performance formula to compute the CPI of the original machine as

$$\text{CPI}_{\text{original}} = .43 \times 1 + .21 \times 2 + .12 \times 2 + .24 \times 2 = 1.57$$

The performance of the original CPU is then given by:

$$\text{CPU time}_{\text{original}} = 1.57 \times \text{IC}_{\text{original}} \times \text{Clock cycle time}_{\text{original}} \tag{3}$$

To compute the CPI of the new CPU, observe that 25% of ALU instructions become register-memory instructions. So, there are $(.25 \times .43)$ fewer ALU operations, $(.25 \times .43)$ fewer loads (since all the loads associated with the removed ALU operations also disappear), and $(.25 \times .43)$ new register-memory ALU instructions. The new instruction count is $.25 \times .43$ smaller than the old one, that is, the new instruction count is $1 - .25 \times .43$. The updated instruction mix as well as the frequency of instructions is given in Table 2. Branches take 3 cycles in the new CPU. Calculating

Table 2: The new instruction mix, frequencies, and CPIs of individual instructions

Operation	Frequency	CPI
ALU Operations	$\frac{.43-.25\times.43}{1-.25\times.43}$	1
Loads	$\frac{.21-.25\times.43}{1-.25\times.43}$	2
Stores	$\frac{.12}{1-.25\times.43}$	2
branches	$\frac{.24}{1-.25\times.43}$	**3**
Register-memory	$\frac{.25\times.43}{1-.25\times.43}$	2

the CPI as before, we get

$$\text{CPI}_{\text{new}} = \frac{(.43 - .25 \times .43) \times 1 + (.21 - .25 \times .43) \times 2 + .12 \times 2 + .24 \times 3 + .25 \times .43 \times 2}{1 - .25 \times .43}$$

$$\text{CPI}_{\text{new}} = 1.908$$

The performance of the new CPU is then given by:

$$\begin{aligned}\text{CPU time}_{\text{new}} &= \text{IC}_{\text{new}} \times 1.908 \times \text{Clock cycle time}_{\text{original}} \\ &= (.893 \times \text{IC}_{\text{original}}) \times 1.908 \times \text{Clock cycle time}_{\text{old}} \\ &= 1.703 \times \text{IC}_{\text{original}} \times \text{Clock cycle time}_{\text{original}} \end{aligned} \quad (4)$$

Comparing (3) and (4), the answer to the question is no: it is a bad idea to add register-memory instructions, because they do not offset the increased execution time of slower branches.

Example 3: Suppose we have made the following measurements:

- Frequency of floating point (FP) operations = 25 %
- Average CPI of FP operations = 4
- Average CPI of other instructions = 1.33
- Frequency of FP square root or FPSQR = 2%
- CPI of FPSQR = 20

Assume that the two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5.Compare these two design alternatives using the CPU performance equation.

Let us obtain the average CPI of the original design as

$$\text{CPI}_{\text{original}} = .25 \times 4 + .75 \times 1.33 = 2.0$$

If we reduce the CPI of the FPSQR instructions from 20 to 2, the impact on CPI will be

$$\begin{aligned}\text{CPI}_{\text{with new FPSQR}} &= \text{CPI}_{\text{original}} - 0.2(\text{CPI}_{\text{old FPSQR}} - \text{CPI}_{\text{new FPSQR only}}) \\ &= 2.0 - 0.2(20 - 2) \\ &= 1.64\end{aligned}$$

We can compute the CPI when all FP operations are enhanced as

$$\text{CPI}_{\text{new FP}} = .25 \times 2.5 + .75 \times 1.33 = 1.625$$

So the speedup corresponding to the overall FP enhancement is

$$\begin{aligned}\text{Speedup}_{\text{new FP}} &= \frac{\text{CPU time}_{\text{original}}}{\text{CPU time}_{\text{new FP}}} \\ &= \frac{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{original}}}{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{new FP}}} \\ &= \frac{2}{1.625} \\ &= 1.23\end{aligned}$$